Open vSwitch Vxlan Performance Acceleration in Cloud Computing Data Center

Yaohua Yan
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
yyh@bupt.edu.cn

Hongbo Wang
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
hbwang@bupt.edu.cn

Abstract—**Cloud computing is one of the most popular Internet concepts, and many large companies provide cloud services to users. These large companies have built their own data centers to support upper layers of cloud services. To save cost and increase flexibility, SDN and virtualization technologies are widely used in data centers. Open vSwitch is an open source virtual switch that supports the OpenFlow protocol. Open vSwitch can provide a true network connection for all virtual machines running on a single physical server. However, the actual performance of Open vSwitch is not satisfactory. The most prominent issue for Open vSwitch is Vxlan, which is a very important tunneling protocol in the data center. In our test environment, Open vSwitch Vxlan throughput is only 2Gbps. This paper presents a new approach to design and implement Open vSwitch. We will use hardware acceleration method to improve the performance of Open vSwitch Vxlan. Our research results show that our design and implementation can improve performance up to 5 times, and the actual throughput closes to 12Gbps.**

Keywords—Cloud Computing; SDN; Virtualization; Open vSwitch; Vxlan; FPGA

I. INTRODUCTION

Today, the application of Cloud Computing technology is more and more extensive. The computer resources can be obtained and expended at any time [1]. This enhances the ease of resource utilization and management greatly. In order to provide users with convenient, reliable and secure cloud services, many large companies choose to establish a unified, centralized data center. Other small and midsize companies are always opting to rent cloud services from large companies to reduce their own operating costs. For these large companies, in order to reduce the data center's difficulty of management and operating costs effectively, the concepts of virtualization and SDN (Software Defined Network) are widely used in the data center [2].

Virtualization technology can be used on x86 machines to simply create multiple virtual machines with full characters, which obviously reduces the cost of operating the data center [3]. But as the size of the data center expands further, management becomes more difficult. The introduction of SDN can greatly reduce the difficulty of data center management. SDN is programmable, dynamic, adaptable and well-managed network architecture. The key technology OpenFlow enables the network traffic the nimble control, causes the network to take the pipeline to become more intelligent.

Vxlan (Virtual eXtensible Local Area Network) is a kind of two-layer packet with three layers protocol encapsulation technology [4]. Vxlan is more in line with the multi-tenant data center business, so Vxlan protocol in the cloud computing data center is widely used [5].

Based on the above situation, Open vSwitch came into being. Open vSwitch is a multi-tier virtual exchange standard for product level quality under the open source Apache2.0 license [6]. The Open vSwitch codes are written in C language and independent of the platform. It is easy to migrate Open vSwitch to other environments. Open vSwitch also supports GRE, Vxlan, which are commonly used in data centers [7]. But the shortcoming of Open vSwitch is also very obvious. Because Open vSwitch is software, its switching performance is largely dependent on the physical host. Like most software, Open vSwitch will face resource competition, processing delays, interruptions and other issues. Especially in the case of high concurrent traffic, Open vSwitch performance will be apparently reduced. In original Open vSwitch, the designer did not take into account Vxlan, and later in order to comply with the trend the Open vSwitch added the support of Vxlan [8] [9]. This causes Open vSwitch to support Vxlan, but the actual performance is poor. In the 1518-byte packet test, Vxlan throughput is only 2Gbps.

The rest of the paper is organized as follows. In Section II, we analyze current architecture, design and implementation of Open vSwitch. Next, in Section III we present our new design and implementation for Open vSwitch. In Section IV, there are some test data to support our feasibility. Finally, we draw conclusions in Section V.

II. CURRENT DESIGN OPEN VSWITCH

In this section, we first present the architecture of Open vSwitch. Then, we explain how Open vSwitch processes an incoming packet. After that, we show performance issues of Open vSwitch. All of our studies are based on the 2.3.0 version of Open vSwitch.

A. *Architecture of Open vSwitch*

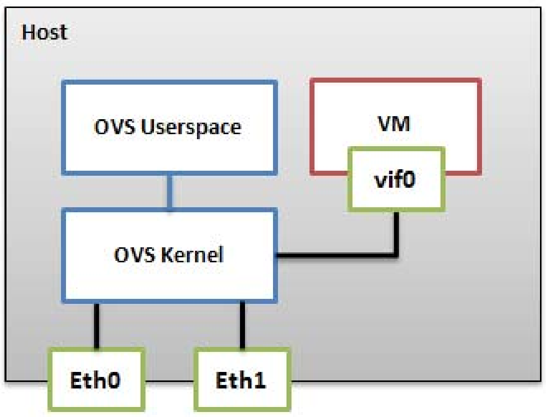

Fig. 1. Artchitecture of Open vSwitch

Open vSwitch is roughly divided into OVS Userspace and OVS Kernel from Figure 1.

OVS Userspace works in the Linux user mode. According to the functionality of the OVS Userspace, its components can be classified into three categories. The first category is the managing tools including ovs-vsctl, ovs-ofctl, ovs-dpctl and ovsdb-clients. These managing tools allow administrator to directly manage Open vSwitch about ports management, flow tables modification, statistic information, database operations and so on. The second category is the database service including ovsdb-server. The ovsdb-server manipulates the database to store switch-level configurations. The third category is the core component including ovs-vswitchd that is responsible for flow table lookups and packet processing. The ovs-vswitchd is the most important component in the OVS Userspace. It modifies the database via the predefined interface and maintains flow tables. In addition, it parses the OpenFlow protocol and exchanges network information with the OpenFlow controller.

OVS Kernel works in the Linux kernel mode and runs as a kernel module. The OVS Kernel components can be classified into two categories. The first category is packet switch. Its functions contain connecting to different devices, analyzing packets sent by devices, looking up the flow table and forwarding packets. The second category is the exchange of control commands from OVS Userspace. Control commands include new flow, delete flow, miss upcall and so on.

Overall, Open vSwitch can be seen as a comprehensive upgrade of the Linux bridge. Their structure, work flows and ideas are almost identical. But Open vSwitch is more versatile and more complex. The biggest difference is that the Linux bridge is a layer 2 switch, but Open vSwitch supports layer 4 switching and OpenFlow.

The main packets exchanging occurs in the OVS Kernel, so we mainly discuss the performance of the OVS Kernel.

B. *Work Flow of Open vSwitch*

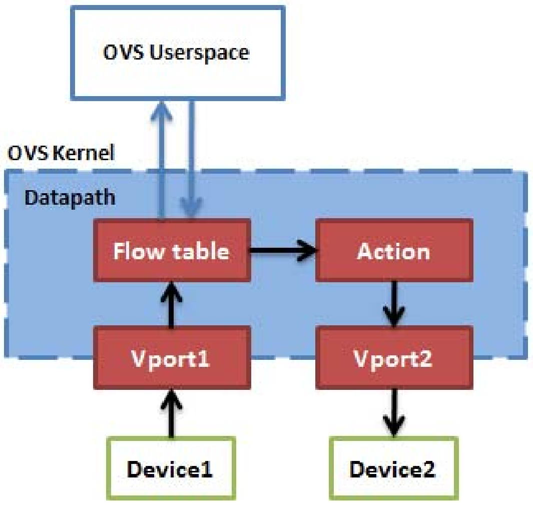

Fig. 2. Work Flow of Open vSwitch

Figure 2 shows the work flow of Open vSwitch. The OVS Kernel mainly includes datapath, flow table, action and vport. A datapath can be regarded as a multi-port router. Vport is same as the physical port of the router, and other devices can be connected to datapath through vport. Vport and the device are a one-to-one relationship. Such one port can only be connected with one device just like the actual physical router. In order to distinguish between different services and protocols, vport have many types, which are shown in Table I.

TABLE I. TYPES OF VPORT

	Type	***Illustration***
1	Netdev	Network interface card
2	Internal	Loop back device
3	Vxlan	Vxlan tunnel
4	Gre	Gre tunnel
5	Gre64	64bit Gre tunnel
6	Lisp	Lisp tunnel

For example, the Netdev vport is used to communicate with network interface card, including the physical interface eth and the virtual interface vif. The Internal vport is similar to the loopback device and is used to communicate with the host. Others like Vxlan type are used for establishing the tunnel.

A router has a routing table that contains many routing entries, and each routing entry consists of a destination ip address, a mask and a next hop. Similarly, the Datapath also has a Flow table that contains a number of flow entries, each of them is made up of a packet key, a mask and an action. The biggest difference between Datapath and router is that Datapath needs to match more keywords to better support the upper OpenFlow protocol. The keywords that need to be matched by the flow table are shown in Table II.

TABLE II. KEYWORDS TO MATCH

	Field	***Keywords***
1	Tunnel	tun_id, ipv4_src, ipv4_dst, tun_flags, ipv4_tos, ipv4_ttl
2	Phy	priority, skb_mark, in_port
3	Mac	mac_src, mac_dst, tci, type
4	Ip/Arp	proto, tos, ttl, frag, ipv4(ip(src, dst), arp(sha， tha)) / ipv6(ip(src, dst, label), nd(target, sll, tll))
5	Tcp/Udp	src_port, dst_port, flags

The matching process of the flow table is basically same as the routing table's. If the result of the packet key AND the mask is equal to the flow key, then the corresponding action (like output, add vlan tag, modify ip head and so on) will be executed. If not, the packet key will try other masks.

After all the masks have been tried, if the packet key can't match any flow, this means that the flow table is missing. The OVS Kernel needs to upload a miss upcall to the OVS Userspace. Then the OVS Userspace will download a new flow to the flow table in OVS Kernel. So the same packets will not miss again.

C. *Vxlan of Open vSwitch*

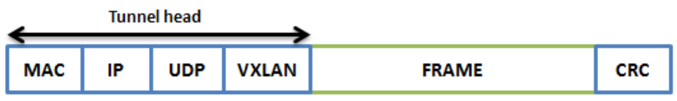

Fig. 3. Vxlan Protocol

Tunnel protocol is one of the key technologies to realize overlay in data center. Vxlan is a layer 3 tunnel protocol. Compared with other tunnel protocols such as GRE, Vxlan is more flexible and more compatible with the current network. Vxlan can be seen as an extension of vlan. Vlan can only identify 4096 networks, which is far from being able to meet the needs of large-scale cloud computing data center. Vxlan's vni is 24-bit, which can identify 16M networks. This can be well adapted to the current needs.

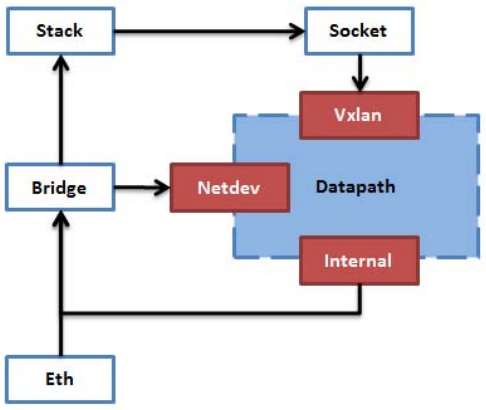

Fig. 4. Receive Vxlan Packet

The Figure 4 shows the work flow of receiving Vxlan packet.

A Vxlan packet goes through the physical network card eth into linux. The bridge module determines whether the packet is received by eth. If so, the packet will be sent to the datapath, or continue to upload to the upper protocol stack. So, the Vxlan packet will go through netdev vport into datapath. But datapath will regard this Vxlan packet as an ordinary packet rather than a Vxlan packet. This means that the handling of this packet is not correct.

However, any packet that first enters the datapath will be missing and will be uploaded to the OVS Userspace via a miss upcall. OVS Userspace can distinguish that it is a Vxlan packet. Because the Vxlan packet has certain characteristics. For example, the outer destination ip is the host's ip address, the outer udp's destination port is 4789 mostly and there is a Vxlan head. The OVS Userspace will download a new flow to the OVS Kernel to make the Vxlan packet to be sent to the No.0 internal vport of the datapath. An internal vport is similar to a loopback device, which causes the Vxlan packet to re-enter linux via a virtual nic. At this time, the Vxlan packet is not received by the eth, so this Vxlan packet will pass the bridge module. Finally, the Vxlan packet will be received by a socket server in the upper protocol stack, then it enters the datapath via Vxlan vport. In this way, Vxlan packet is handled correctly.

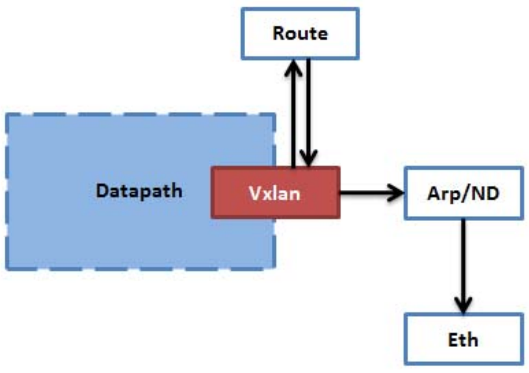

Fig. 5. Send Vxlan Packet

The Figure 5 shows the work flow of sending Vxlan packet.

Vxlan is transparent for users. It is to say that the virtual machine will only receive and send normal packets. The set tunnel action mainly contains the Vxlan vni, the tunnel source ip and the tunnel destination ip. Vxlan vport encapsulates the outer tunnel header for normal packets. Open vSwitch's support on the Vxlan is not complete. Due to the initial design of Open vSwitch, the Vxlan had not been taken into account.

The outer tunnel head mainly includes MAC, IP, UDP and Vxlan. The Vxlan head only needs to be set vni. The UDP head's destination port number is 4789 and the source port number is a random number. They are relatively simple, but IP and MAC head need more complex operations. The set tunnel action offers the tunnel source ip address and the destination ip address, but the tunnel source ip address is always empty. This is understandable, because Vxlan is different from GRE. GRE is a one-to-one tunnel protocol, but Vxlan is one-to-many. The tunnel destination ip (generally called remote ip) is specified when user creates a Vxlan port. But the outer tunnel source ip can only lookup from linux router. The linux router can also return the export interface, so that the outer source mac address and destination mac address can be gotten through the neighbor subsystem (ipv4 corresponding ARP, ipv6 corresponding ND). Now, the encapsulation of this Vxlan packet is completed.

D. Problems Summary

After the above analysis, we can know why the performance of Open vSwitch Vxlan is so poor.

a. In the receiving process, the Vxlan packet needs to enter the datapath twice, and also needs to go through the protocol stack.

b. In the sending process, the packet's encapsulation needs to look up the linux router and the neighbor subsystem, however their performance is poor.

III. OPEN VSWITCH ACCLERATION IN HARDWARE

In this section, we will present the design and implement of Open vSwitch acceleration by hardware, and how we solve the problem of Vxlan in the original Open vSwitch.

A. Overall Idea

The idea of the Open vSwitch hardware acceleration is only to change the OVS Kernel, and using the hardware to accelerate look-up table and forwarding. Meanwhile, we will improve the way of handling the Vxlan packet.

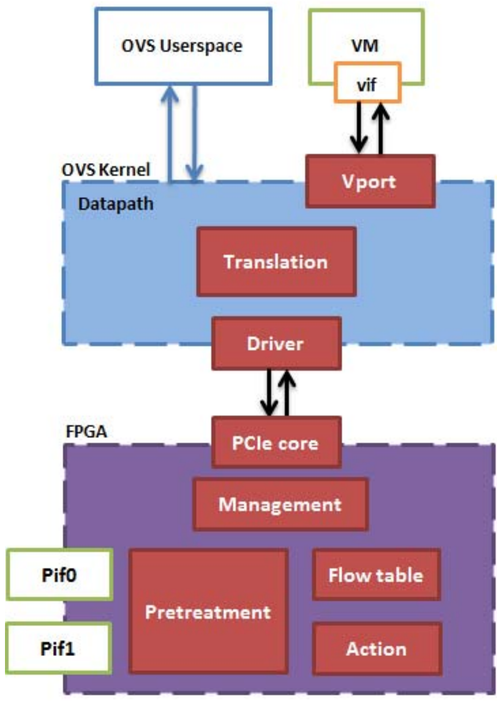

Fig. 6. Artchitecture of Open vSwitch Accleration in Hardware

In OVS Kernel from Figure 6, we completely retain the communication interface between OVS Userspace and OVS Kernel. We also retain the Netdev vport to communicate with virtual machines. So OVS Userspace and virtual machine do not need to do any modification. Compared with the flexiblity of software, in the situation of hardware, data need to be specified length, fixed format and clear meaning. Therefore, we add a Translation module in OVS Kernel to transform the commands and data between hardware and software.

Our FPGA plate model is "virtex-7 FPGA gen3 integrated block for PCI Express V3.0" with two 10G optical modules and overall just looks like a nic. The Driver's design and development is also in accordance with the network card model. On the FPGA board, the PCIe core realizes the PCIe interaction, and the Management is used to distinguish between different messages.

The packet sent by the virtual machines or received by the 10G optical modules will all enter the Pretreatment module. The Pretreatment module will parse the packet and extract the keywords to form a packet key. Then it will enter the Flow table for the query.

In addition to some differences, the working principle is basically consistent with the original Open vSwitch. Next we will focus on the improvement of Vxlan.

B. Improvement of Vxlan

According to our previous analysis, the reason why performance of the original Open vSwitch Vxlan is so poor is the operations of receiving and sending. So we mainly improve these two operations.

When a packet goes through the Pif into the Pretreatment module, first of all, we distinguish whether it is a Vxlan packet. If it is, the packet will remove the outer Vxlan tunnel encapsulation, then we can get the original mac frame. This frame is treated as other ordinary packet to look up flow table and forwarding. In this way, we can avoid entering datapath twice in the original Open vSwitch.

When an original packet needs to be encapsulated as a Vxlan packet, we design an IP-MAC Table in Pretreatment module to replace the original linux router and neighbor subsystem.

TABLE III. IP-MAC TABLE

	Remote Ip	***Local Ip***	***Dst Mac***	***Out Port***
1	10.1.1.1	10.1.2.2	a1:bf:ce:dd:02:11	eth0
2	10.1.1.2	10.1.2.2	a1:bf:c5:08:34:5d	eth0
3	…	…	…	…

As shown in Table III, the IP-MAC Table's working principle is based on ARP Learning. When receiving an arp request packet through the pif, the source ip, destination ip, destination mac and interface number will be respectively written in the IP-MAC Table's remote ip, local ip, dst mac and out port. Among them, remote ip is the primary key of the IP-MAC Table. If we want to encapsulate a Vxlan packet, it just needs to look up in the IP-MAC Table by tunnel destination ip what the set tunnel action includes.

IV. Experimental Verification

In this section, we mainly demonstrate our experimental environment and experimental data.

A. *Experimental Environment*

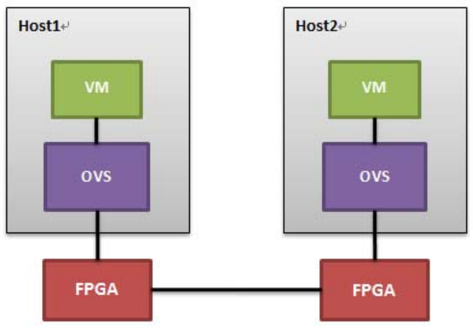

Fig. 7. Experimental Environment

Vxlan is a type of tunnel protocol, so we use two hosts in Figure 7. In order to avoid other interference factors, the two hosts are connected directly with each other. And in these two hosts, their configurations and settings are all the mirrored.

B. *Experimental Data*

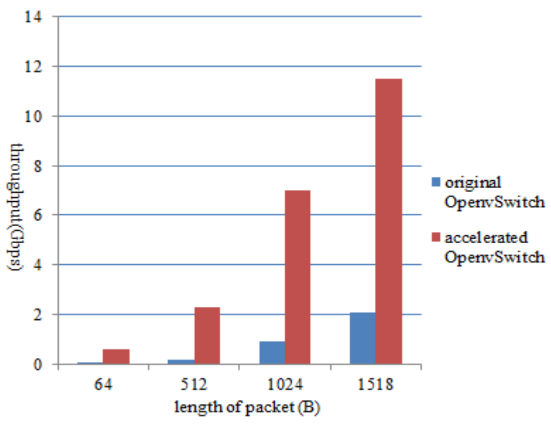

Fig. 8. Experimental Data

Compared to the original Open vSwitch, the throughput of Vxlan has been improved very obviously, which is shown in Figure 8. However, the rate of the original Open vSwitch and the accelerated Open vSwitch is decreasing very apparently, with the length of the packet increases.

V. Conclusion

In this paper, we studied the design and implementation of original Open vSwitch and identified its performance bottleneck when it exchanges Vxlan packets. We proposed a new design and implementation to reduce the required operations in the receiving process and the sending process. The performance evaluation results show that our new design and implementation can improve the performance by up to 5 times, and the actual throughput close to 12Gbps.

Acknowledgment

This work is supported by the Fundamental Research Funds for the Central Universities (No. 2013RC1104); the National Natural Science Foundation of China (No. 61002011); the 863 Program of China (No. 2013AA013303).

References

[1] Sá T T, Soares J M, Gomes D G. CloudReports: uma ferramenta gráfica para a simulaç ao de ambientes computacionais emnuvem baseada no framework CloudSim[J]. 2011.

[2] T. Anderson, L. Peterson, S. Shenker and J. Turner, Overcoming the Internet impasse through virtualization, in Computer, vol. 38, no. 4, pages 34-41, April 2005.

[3] "Software-Defined Networking: The New Norm for Networks," a white paper of Open Networking Foundation, April 13, 2012.

[4] Dennis Cai, Sai Natarajan, "The Evolution of the Carrier Cloud Networking", Service Oriented System Engineering (SOSE), 2013

[5] Ryota Kawashima, Hiroshi Matsuo, "Non-tunneling Edge-Overlay Model Using OpenFlow for Cloud Datacenter Networks", IEEE 5th International Conference on Cloud Computing, 2013

[6] Open vSwitch, available at http://http://Open vSwitch.org/

[7] Vxlan in Data Centre network at http://forum.h3c.com/thread-115802-1-1.html

[8] Ibáñez, Bart De Schuymer, Jad Naous, Diego Rivera, Elisa Rojas, "Implementation of ARP-path low latency bridges in Linux and OpenFlow/NetFPGAGuillermo", Juan A. Carral, 2011

[9] Franco Callegati, Walter Cerroni, Chiara Contoli, Giuliano Santandrea, "Performance of Network Virtualization in cloud computing infrastructures: The OpenStack case", 2014